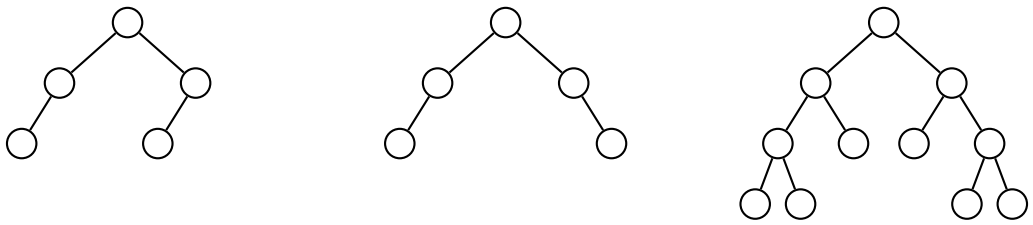
Estructuras de Datos y Algoritmos

Grados de la Facultad de Informática (UCM)

Examen de SEPTIEMBRE, Segundo Cuatrimestre, 6 de septiembre de 2018

1. (2.5 puntos) Un árbol binario no vacío es *simétrico* respecto al eje vertical que pasa por la raíz si al "doblarlo" por ese eje todo nodo de un lado coincide con un nodo del otro. Por ejemplo, de los siguientes árboles, el de la izquierda no es simétrico pero los otros dos sí lo son.



Completa el código del fichero `main1parcial.cpp` con la implementación de una función que, dado un árbol binario no vacío, averigüe si es simétrico o no.

2. (2.5 puntos) Implementa una función que dada una cola de números enteros que está ordenada crecientemente según el valor absoluto de sus elementos, la modifique de forma que quede ordenada crecientemente según el valor de sus elementos. Puedes utilizar estructuras de datos auxiliares; justifica tu elección.

Por ejemplo, si la cola contiene los elementos $-1, 1, -3, 4, 5, -7, 9, 10, -15$, al final debe contener los elementos $-15, -7, -3, -1, 1, 4, 5, 9, 10$.

Completa el código del fichero `main2parcial.cpp` con la implementación de esta función.

3. (3 puntos) Queremos extender la clase `deque` (lo importante para el ejercicio es que internamente la cola está representada con una lista enlazada circular doble de nodos dinámicos con nodo fantasma) con una nueva operación *engordar* que añada a una lista enlazada doble el contenido de otra lista enlazada doble dada de la siguiente forma: los nodos de la segunda lista se colocarán alternativamente al principio y al final de la primera lista. La lista recibida como argumento pasará a ser vacía.

Para resolver este ejercicio no se puede crear ni destruir memoria dinámica (hacer `new` o `delete`), ni tampoco modificar los valores almacenados en las listas enlazadas.

Completa el código del fichero `main3.cpp` con la implementación de esta nueva operación.

4. (2 puntos) Dada una secuencia de números enteros positivos (no necesariamente ordenados y con posibles repeticiones) y un rango $[inf, sup]$, se quieren imprimir en orden creciente todos los números del rango que no estén en la secuencia. Para ello se pide implementar una función `enRangoYNoEnSecuencia` que dada la secuencia de entrada `sec` como `list<int>`, y los enteros `inf` y `sup`, devuelva la secuencia de salida, también como `list<int>`, con los números del rango que no estén en `sec`, ordenados crecientemente. Se valorará el coste del algoritmo implementado, el cual debes indicar y justificar. Aclaración: Puedes usar TADs auxiliares. Ejemplo: dada la secuencia `3 9 1 3 7 2` y el rango `[3,8]` se debe imprimir la secuencia `4 5 6 8`.

Completa el código del fichero `main4.cpp` con la implementación de esta operación.

Normas de realización del examen

1. Debes programar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc/domjudge/team`.

2. Escribe comentarios que expliquen tu solución, justifiquen por qué se ha hecho así y ayuden a entenderla. Calcula la complejidad de todas las funciones que implementes.

3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. El nombre de usuario y contraseña que has estado utilizando durante la evaluación continua **no** son válidos.

4. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que entregues.

5. Descarga el fichero `http://exacrc/Docum122448.zip` que contiene material que debes utilizar para la realización del examen (implementación de las estructuras de datos, ficheros con código fuente y ficheros de texto con algunos casos de prueba de cada ejercicio del enunciado).

6. Si la necesitas, puedes encontrar ayuda sobre la librería estándar de C++ en `http://exacrc/cpp/reference/en/`.

7. Tus soluciones serán evaluadas por el profesor independientemente del veredicto del juez automático. Para ello, el profesor tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.